

Software Design, Modelling and Analysis in UML

Lecture 09: Class Diagrams III

2014-11-25

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany

– 09 – 2014-11-25 – main –

Contents & Goals

Last Lectures:

- completed class diagrams... except for associations

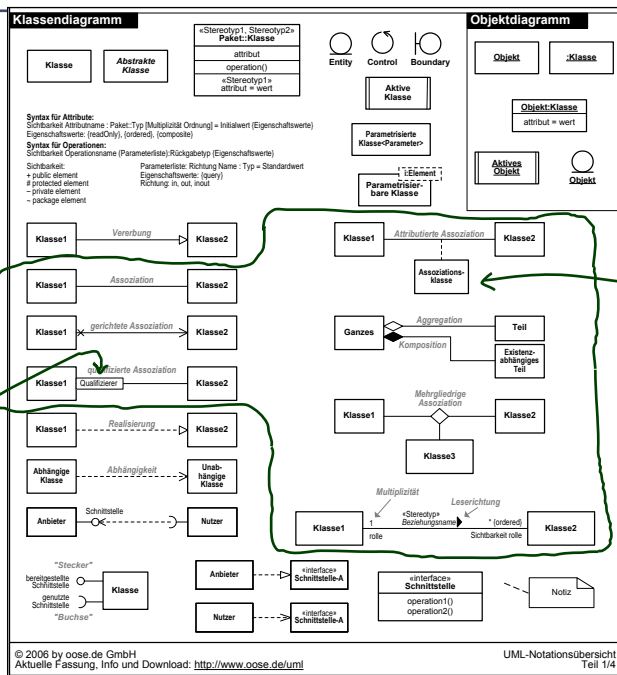
This Lecture:

- **Educational Objectives:** Capabilities for following tasks/questions.
 - Please explain this class diagram with associations.
 - Which annotations of an association arrow are semantically relevant?
 - What's a role name? What's it good for?
 - What is "multiplicity"? How did we treat them semantically?
 - What is "reading direction", "navigability", "ownership", ...?
 - What's the difference between "aggregation" and "composition"?
- **Content:**
 - Study concrete syntax for "associations".
 - (**Temporarily**) extend signature, define mapping from diagram to signature.
 - Study effect on OCL.
 - Btw.: where do we put OCL constraints?

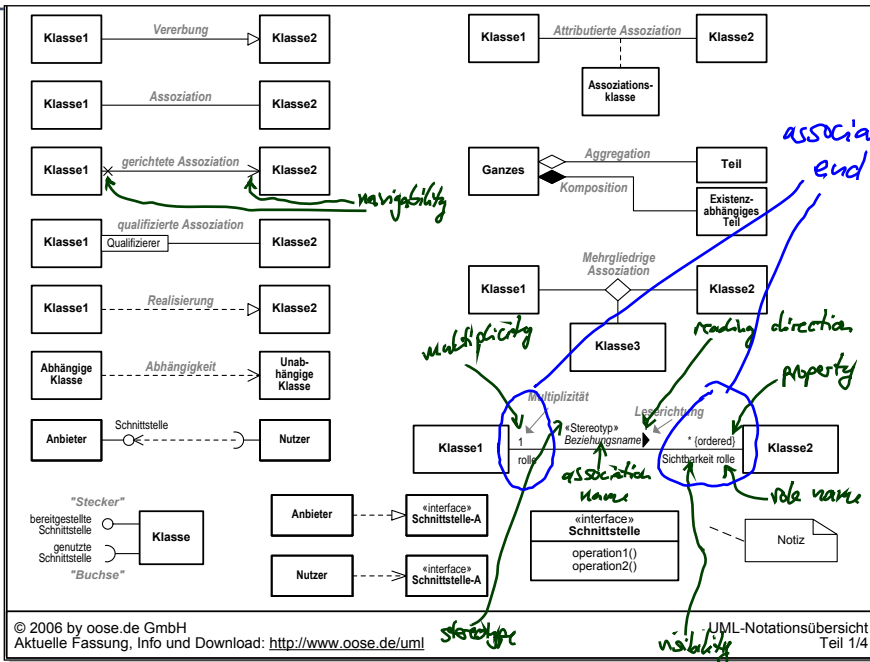
– 09 – 2014-11-25 – Prelim –

Associations: Syntax

UML Class Diagram Syntax [Oestereich, 2006]



UML Class Diagram Syntax [Oestereich, 2006]



- 09 - 2014-11-25 - Sassocsyn -

UML Class Diagram Syntax [OMG, 2007b, 61;43]

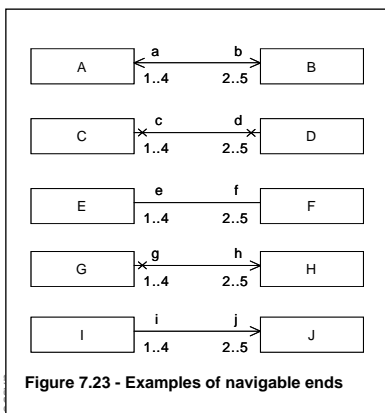


Figure 7.23 - Examples of navigable ends

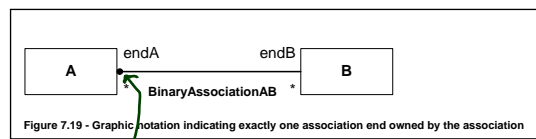


Figure 7.19 - Graphic notation indicating exactly one association end owned by the association

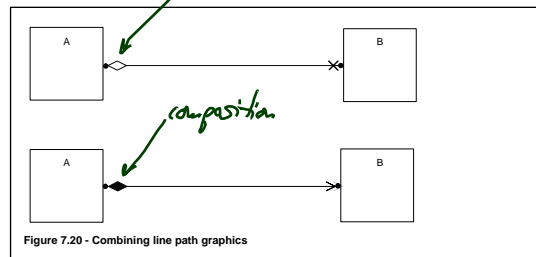


Figure 7.20 - Combining line path graphics

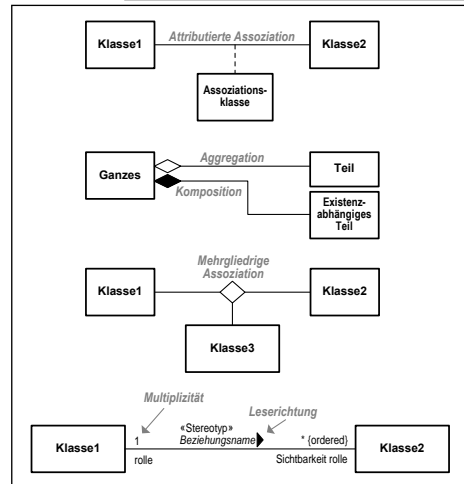
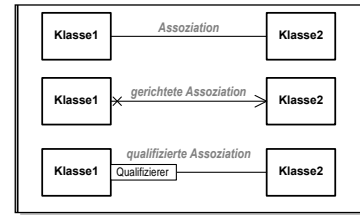
- 09 - 2014-11-25 - Sassocsyn -

What Do We (Have to) Cover?

An **association** has

- ! • a **name**,
 - ✓ • a **reading direction**, and
 - ! • at least two **ends**.
 - ! • a **set of stereotypes** *just a hint to the readers of the diagram*
- Each **end** has
- ! • a **role name**,
 - ! • a **multiplicity**,
 - ! • a set of **properties**, such as **unique**, **ordered**, etc.
 - • a **qualifier**, (not in lecture)
 - ! • a **visibility**,
 - ! • a **navigability**,
 - ! • an **ownership**,
 - ! • and possibly a **diamond**. (exercises)

Wanted: places in the signature to represent the information from the picture.



- 09 - 2014-11-25 - Sassocsyn -

7/44

(Temporarily) Extend Signature: Associations

Only for the course of Lectures 9/10 we assume that each attribute in V

- **either** is $\langle v : \tau, \xi, expr_0, P_v \rangle$ with $\tau \in \mathcal{T}$ (as before),
- **or** is an **association** of the form

$$\langle r : \langle role_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle, \dots, \langle role_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle$$

↑
assoc. name

↑
assoc. end

the class where this association end is located

- 09 - 2014-11-25 - Sassocsyn -

8/44

(Temporarily) Extend Signature: Associations

Only for the course of Lectures 9/10 we assume that each attribute in V

- **either** is $\langle v : \tau, \xi, expr_0, P_v \rangle$ with $\tau \in \mathcal{T}$ (as before),
- **or** is an **association** of the form

$$\langle r : \begin{array}{l} \langle role_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle, \\ \vdots \\ \langle role_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \end{array} \rangle$$

where

- $n \geq 2$ (at least two ends),
- $r, role_i$ are just **names**, $C_i \in \mathcal{C}$, $1 \leq i \leq n$,
- the **multiplicity** μ_i is an expression of the form

$$\mu ::= * \mid N \mid N..M \mid N..* \mid \mu, \mu \quad (N, M \in \mathbb{N})$$

- P_i is a set of **properties** (as before),
- $\xi \in \{+, -, \#, \sim\}$ (as before),
- $\nu_i \in \{\times, -, >\}$ is the **navigability**,
- $o_i \in \mathbb{B}$ is the **ownership**.

8/44

- 09 - 2014-11-25 - Sassocsyn -

(Temporarily) Extend Signature: Associations

Only for the course of Lectures 9/10 we assume that each attribute in V

- **either** is $\langle v : \tau, \xi, expr_0, P_v \rangle$ with $\tau \in \mathcal{T}$ (as before),
- **or** is an **association** of the form

$$\langle r : \begin{array}{l} \langle role_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle, \\ \vdots \end{array} \rangle$$

Alternative syntax for multiplicities:

$$\mu ::= N..M \mid N..* \mid \mu, \mu \quad (N, M \in \mathbb{N} \cup \{*\})$$

- and define $*$ and N as abbreviations.
- **Note:** N could abbreviate $0..N$, $1..N$, or $N..N$. We use last one.

$$\mu ::= * \mid N \mid N..M \mid N..* \mid \mu, \mu \quad (N, M \in \mathbb{N})$$

- P_i is a set of **properties** (as before),
- $\xi \in \{+, -, \#, \sim\}$ (as before),
- $\nu_i \in \{\times, -, >\}$ is the **navigability**,
- $o_i \in \mathbb{B}$ is the **ownership**.

8/44

- 09 - 2014-11-25 - Sassocsyn -

(Temporarily) Extend Signature: Basic Type Attributes

Also only for the course of this lecture

- we only consider **basic type attributes** to “belong” to a class (to appear in $atr(C)$),
- associations** are not “owned” by a particular class (do not appear in $atr(C)$), but live on their own.

Formally: we only call

$$(\mathcal{I}, \mathcal{C}, V, atr)$$

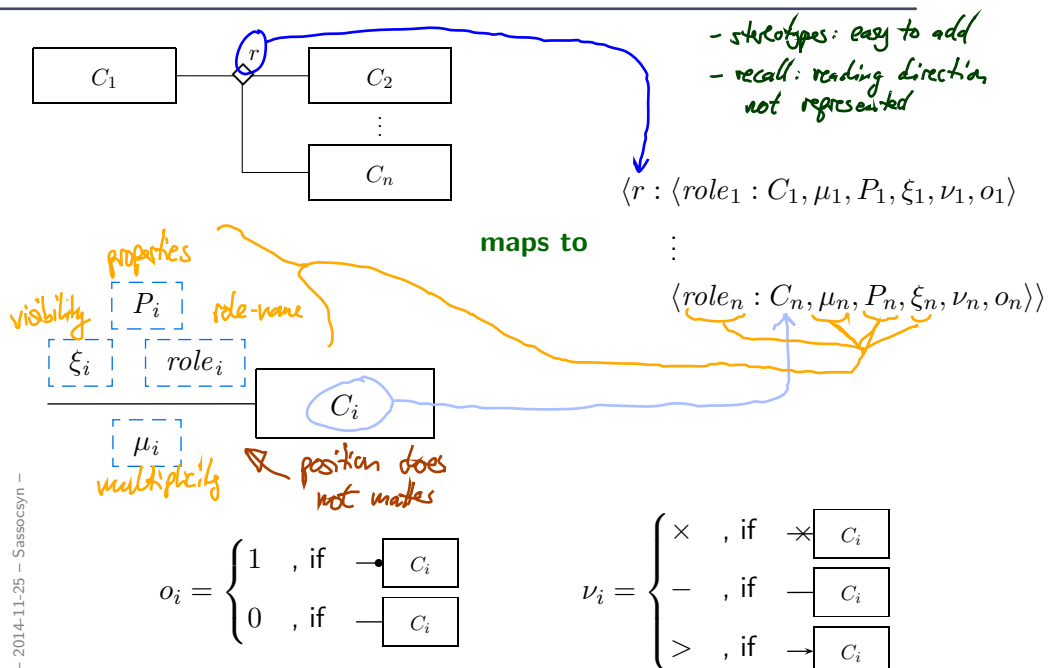
a **signature (extended for associations)** if

$$atr : \mathcal{C} \rightarrow 2^{\{v \in V \mid v: \tau, \tau \in \mathcal{T}\}}.$$

- 09 - 2014-11-25 - Sassocsyn -

9/44

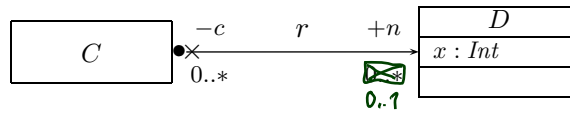
From Association Lines to Extended Signatures



- 09 - 2014-11-25 - Sassocsyn -

10/44

Association Example



Signature:

$$\mathcal{S} = \left(\{Int\}, \{C, D\}, \{ \langle x: Int, +, *, \emptyset \rangle, \right.$$

$$\left. \langle r: \langle C: C, 0..*, \emptyset, -, X, 1 \rangle, 0 \rangle, \right.$$

$$\left. \langle n: D, 0..1, \emptyset, +, >, 0 \rangle \right\}$$

$$\left. \left. \begin{array}{l} C \mapsto \emptyset \\ D \mapsto \{x\} \end{array} \right\} \right)$$

← only basic type attributes here

What If Things Are Missing?

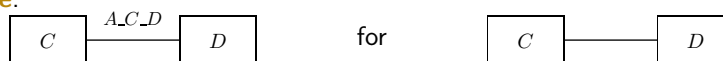
Most components of associations or association end may be omitted. For instance [OMG, 2007b, 17], Section 6.4.2, proposes the following rules:

- **Name:** Use

$$A_{\langle C_1 \rangle \dots \langle C_n \rangle}$$

if the name is missing.

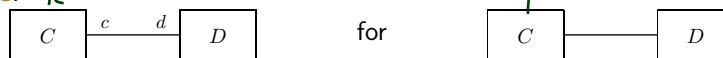
Example:



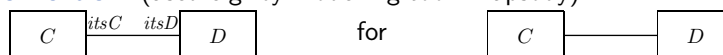
- **Reading Direction:** no default.

- **Role Name:** use the class name at that end in lower-case letters

Example:



Other convention: (used e.g. by modelling tool Rhapsody)



What If Things Are Missing?

- **Multiplicity:** 1

In my opinion, it's safer to assume 0..1 or * if there are no fixed, written, agreed conventions ("expect the worst").

- **Properties:** \emptyset
- **Visibility:** public
- **Navigability and Ownership:** not so easy. [OMG, 2007b, 43]

"Various options may be chosen for showing navigation arrows on a diagram.

In practice, it is often convenient to suppress some of the arrows and crosses and just show exceptional situations:

- *Show all arrows and x's. Navigation and its absence are made completely explicit.*
- *Suppress all arrows and x's. No inference can be drawn about navigation.*
This is similar to any situation in which information is suppressed from a view.
- *Suppress arrows for associations with navigability in both directions, and show arrows only for associations with one-way navigability.*

In this case, the two-way navigability cannot be distinguished from situations where there is no navigation at all; however, the latter case occurs rarely in practice."

Wait, If Omitting Things...

- **...is causing so much trouble** (e.g. leading to misunderstanding), why does the standard say "**In practice, it is often convenient...**"?

Is it a good idea to trade **convenience** for **precision/unambiguity**?

It depends.

- Convenience as such is a legitimate goal.
- In UML-As-Sketch mode, precision "doesn't matter", so convenience (for writer) can even be a primary goal.
- In UML-As-Blueprint mode, **precision** is the **primary goal**. And misunderstandings are in most cases annoying.

But: (even in UML-As-Blueprint mode)

If all associations in your model have multiplicity *, then it's probably a good idea not to write all these *'s.

So: tell the reader about it and leave out the *'s.

Association Semantics

Overview

What's left? **Named** association with at least two typed **ends**, each having

- a **role name**,
- a set of **properties**,
- a **navigability**, and
- a **multiplicity**,
- a **visibility**,
- an **ownership**.

The Plan:

- Extend **system states**, introduce so-called **links** as instances of associations — depends on **name** and on **type** and **number** of ends.
- Integrate **role name** and **multiplicity** into **OCL syntax/semantics**.
- Extend **typing rules** to care for **visibility** and **navigability**
- Consider **multiplicity** also as part of the **constraints** set $Inv(CD)$.
- **Properties**: for now assume $P_v = \{\text{unique}\}$.
- **Properties** (in general) and **ownership**: later.

Association Semantics: The System State Aspect

Associations in General

Recall: We consider associations of the following form:

$$\langle r : \langle role_1 : C_1, \mu_1, P_1, \xi_1, \nu_1, o_1 \rangle, \dots, \langle role_n : C_n, \mu_n, P_n, \xi_n, \nu_n, o_n \rangle \rangle$$

Only these parts are relevant for extended system states:

$$\langle r : \langle role_1 : C_1, -, P_1, -, -, - \rangle, \dots, \langle role_n : C_n, -, P_n, -, -, - \rangle \rangle$$

(recall: we assume $P_1 = P_n = \{\text{unique}\}$).

The UML standard thinks of associations as **n-ary relations** which “**live on their own**” in a system state.

That is, **links** (= association instances)

- **do not** belong (in general) to certain objects (in contrast to pointers, e.g.)
- are “first-class citizens” **next to objects**,
- are (in general) **not** directed (in contrast to pointers).

Links in System States

$$\langle r : \langle role_1 : C_1, -, P_1, -, -, - \rangle, \dots, \langle role_n : C_n, -, P_n, -, -, - \rangle \rangle$$

Only for the course of Lectures 9/10 we change the definition of system states:

Definition. Let \mathcal{D} be a structure of the (extended) signature $\mathcal{S} = (\mathcal{I}, \mathcal{C}, V, atr)$.

A **system state** of \mathcal{S} wrt. \mathcal{D} is a pair (σ, λ) consisting of

- a type-consistent mapping

$$\sigma : \mathcal{D}(\mathcal{C}) \mapsto (atr(\mathcal{C}) \mapsto \mathcal{D}(\mathcal{I})),$$

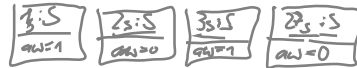
values for basic types only
- a mapping λ which assigns each association $\langle r : \langle role_1 : C_1, \dots, role_n : C_n \rangle \rangle \in V$ a relation

$$\lambda(r) \subseteq \mathcal{D}(C_1) \times \dots \times \mathcal{D}(C_n)$$

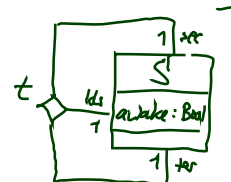
(i.e. a set of type-consistent n -tuples of identities).

- 09 - 2014-11-25 - Sassocsem -

Example



$\langle t : \langle k_1 : s \rangle, \langle sec : s \rangle, \langle tr : s \rangle \rangle$



$$\sigma = \{ 1s \mapsto \{ attr \mapsto 1 \}, 2s \mapsto \{ attr \mapsto 0 \}, 3s \mapsto \{ attr \mapsto 1 \}, 2_2s \mapsto \{ attr \mapsto 0 \} \}$$

$$\lambda = \{ t \mapsto \left\{ \begin{array}{l} (1s, 2s, 3s), \\ (1s, 2_2s, 3s), \\ (2s, 5s, 6s), \\ (2s, 3s, 1s), \\ (3s, 3s, 3s) \end{array} \right\} \}$$

students may join multiple teams

links may also have dangling references

one student may assume all roles (add an OCL constraint if not desired)

Object Diagrams:

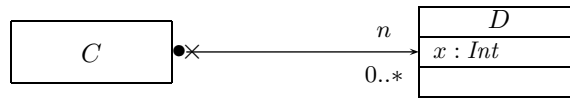


roles?

- we need hyperset in general (not best with role labels)

WE WILL NOT FORMALLY DEFINE THAT

Association/Link Example



Signature:

$$\mathcal{S} = (\{Int\}, \{C, D\}, \{x : Int, \langle A_C_D : \langle c : C, 0..*, +, \{\text{unique}\}, \times, 1 \rangle, \langle n : D, 0..*, +, \{\text{unique}\}, >, 0 \rangle \rangle\}, \{C \mapsto \emptyset, D \mapsto \{x\}\})$$

A **system state** of \mathcal{S} (some reasonable \mathcal{D}) is (σ, λ) with:

$$\sigma = \{1_C \mapsto \emptyset, 3_D \mapsto \{x \mapsto 1\}, 7_D \mapsto \{x \mapsto 2\}\}$$

$$\lambda = \{A_C_D \mapsto \{(1_C, 3_D), (1_C, 7_D)\}\}$$

Extended System States and Object Diagrams

Legitimate question: how do we represent system states such as

$$\sigma = \{1_C \mapsto \emptyset, 3_D \mapsto \{x \mapsto 1\}, 7_D \mapsto \{x \mapsto 2\}\}$$

$$\lambda = \{A_C_D \mapsto \{(1_C, 3_D), (1_C, 7_D)\}\}$$

as **object diagram**?

References

— [Oestereich, 2006] Oestereich, B. (2006). *Analyse und Design mit UML 2.1, 8. Auflage*. Oldenbourg, 8. edition.

[OMG, 2007a] OMG (2007a). Unified modeling language: Infrastructure, version 2.1.2. Technical Report formal/07-11-04.

[OMG, 2007b] OMG (2007b). Unified modeling language: Superstructure, version 2.1.2. Technical Report formal/07-11-02.